Project Ideas for the George Varghese Espresso Prize

To be eligible for the George Varghese Espresso Prize, your project should first satisfy the original *Router Implementation* project specifications mentioned in the course site. While it is not necessary to receive a perfect score on the standard assignment, the basic functionality must be generally correct.

If you are interested in competing for the prize you should extend your router implementation to do something cool!☺ What it does is totally up to you, although you might want to use this opportunity to explore some of the concepts we discussed in class. To provide some inspiration, we list some potential extension below; you are welcome to do any or all of these.

The prize will be awarded to the submission (or submissions) that, in the judgment of the instructor and the TAs, best demonstrates excitement and enthusiasm for the material in CSE123 through implementing extended routing features. One way to do so is to implement one or more of the extensions below. Should multiple projects receive perfect scores, the prize will go to the project that has implemented the most additional functionality in an elegant fashion. Multiple awards may be made at the sole discretion of the Instructor. The winner(s) will be announced at the final exam.

Queuing and Scheduling:

Routers usually implement some sort of queuing and scheduling mechanisms (ranging from the simple FIFO to more complex methods), that manage how the packets are buffered (and eventually scheduled/transmitted) while waiting to be transmitted. These mechanism in turn control how much bandwidth is allocated to a certain packet. You are going to implement:

1. Queuing system for packets on the outgoing interfaces.
2. Segregation of Flows into either a 'High Priority Queue (HPQ)' or a 'Low Priority Queue (LPQ)' based on their TOS field.
3. Flow tracking/counting the number of active flows based on the TCP SYN/FIN flags.
4. Scheduling packets onto the networks based on the number of active flows and priority.

Implementation Requirements:

Queuing and Segregation: You have to maintain two Queues (HPQ and LPQ) for each interface. In this design a HPQ should get 10x more bandwidth than LPQ. After a packet is processed by the router and ready to be sent it is queued in one of the Queues defined above based on the TOS field. E.g: if the TOS = 0x02 then it is placed in the LPQ and if the TOS = 0x04 it is placed in the HPQ. Flow tracking/counting: You can count the number of active flows by accessing the TCP SYN/FIN flags. You should maintain two flow counters for counting the number of active flows in each queue

and every time you receive a SYN flag you will increment the counter and every time you receive a FIN flag you will decrement the flow counter.

Triton Scheduler: Packet scheduling is a key design consideration when multiple buffers are involved. We will be working on weighted processor sharing based on the information in the flow counter and the priority of the Queues (remember HPQ should get 10x more bandwidth than LPQ). E.g: if the flow ratio of HPQ and LPQ is 1 then for every 1 packets serviced from the LPQ, 10 should be serviced from the HPQ.
It Determines:

1) which packets get transmitted.
2) when packets get transmitted.

Classifier: assigns packets to different queues (Based on TOS).
Triton Scheduler: selects packets to be transmitted from queues (Based on number of flows and type of queue).

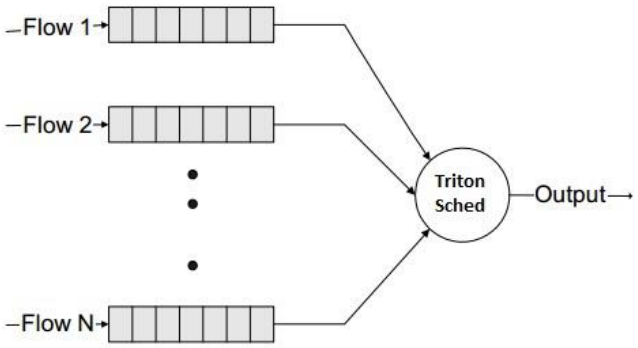

Figure 1: Triton Scheduler Connected to N number of Queues

References:
http://en.wikipedia.org/wiki/Deficit_round_robin
Peterson & Davie Chapter6-Pages(492-499)

Shaping Traffic:

You are required to control the amount of traffic coming in or going out of its interfaces. In this task you are going to filter packets based on their IP addresses and fulfill certain performance requirements give to you in a file.
e/g: Rate limiting by not allowing one of the senders to send more than a certain amount of data in a certain time window. (The senders can also be limited to send a maximum amount of data (50 MB) on eth0.)

We are going to be using source IP and destination IP address to define traffic rules in our setup. I am going to refer to Source IP as “srcIP” and Destination IP as “dstIP” from now on

Actions/Operations: **rate_out, rate_in, data_out, data_in**

rate_out: The rate of outgoing traffic, from the router, destined for dstIP (If only dstIP is provide). OR, the rate of outgoing traffic, from the router, to a dstIP, originating from a specific IP (source IP).

rate_in: The rate of incoming traffic, to the router, from a srcIP (If only srcIP is provide). OR, the rate of incoming traffic, to the router, from srcIP and destined for dstIP (If bothe srcIP and dstIP are provided).

data_out: The amount of data that a dstIP (If only dstIP is provide) is allowed to receive from an interface of the router.

data_in: The amount of data that a srcIP (If only srcIP is provide) is allowed to transmit through an interface of the router.

Data/Rate value field: If the operation is Data assume units MB, else if the operation is Rate assume units MB/s.

General Format of operation instructions:
<action>from<incoming IP block>to<outgoing IP block> <Data/Rate value field >

Eg: data_out from 127.0.0.0/8 to any 50 • This means that all the IP packets from 127.0.0.0/8 are allowed to send only 50 MB of data (through the router).

Your router should be able to take a file as an input containing a set of instructions (traffic rules in the format described earlier) and configure its interfaces according to that.

Firewall :

A firewall is a hardware or software security system that inspects packets flowing to and from the network device, checks compliance of each packet against a set of rules, and decides if the packet can be sent through to the destination. There are three levels in which a firewall can operate:

1. Packet Filter • Based on the contents of the packet headers.
2. Stateful inspection • Based on the state of a TCP connection.
3. Application Layer Filters • Based on the application layer information carried by each packet.

Out of the three mentioned above, you should start with the first type • Packet Filter.

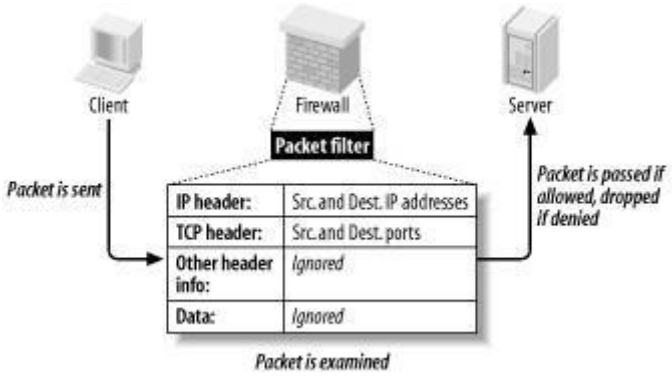

Image from http://eduunix.ccut.edu.cn

Assume that all the clients in your network topology belong to a student organization named *Triton*. Triton wants to safeguard its network from malicious requests from the internet. You, the owner of Triton's edge router, have negotiated with Triton and have come to an understanding that Triton's board will give you a set of filtering rules in the following format:

<action> <protocol> from <incoming IP block> [<optional incoming port>] to <outgoing IP block> [<optional outgoing port>] [<optional direction>]

Direction (can be "in" or "out") is specified with respect to the way the connection is first established. For example, an "inbound" flow is one whose first packet comes from <incoming IP block>.

Eg: deny ip from 127.0.0.0/8 to any • This means that all the IP packets from 127.0.0.0/8 directed to any IP address should be rejected by the router.

Triton's board has kindly accepted to give the firewall rules in the form of a file which your router should read to create its firewall rules. For example, the firewall spec file can look like the following (assuming Triton's network has a network address of 20.0.0.0/8).

deny tcp from 20.0.0.0/8 to any 22 out
// DROP outgoing SSH connections from Triton network to the public network.

allow tcp from any to 20.0.0.0/8 80 in
// ALLOW HTTP access from public network to Triton network.

deny tcp from any to 20.0.0.0/8 in
// DROP all other inbound TCP connections from public network to Triton network.

allow tcp from any to any
// ALLOW all other TCP connections.

You should apply these rules in order. In other words, if a packet matches more than one rule, you should apply the action specified by the first rule matched.

Your router should be able to change its firewall policies based on Triton board's specification no later than 1 second after the change was made to the firewall spec file.

Further information regarding firewall operation can be found in the following links:

http://en.wikipedia.org/wiki/Firewall_(computing)
http://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8
Chapter 8.5 in P&D.

You are welcome to implement additional firewall functionality; such as stateful transport or application•layer packet inspection. Please document any extensions required to the configuration language to support your features in a README.

Network Address Translation (NAT):

NAT is a way to spoof all traffic going in and out of a network as though they are originating from and destined to a single IP address. For example, most home networks are set up such that the cable/DSL modem has one IP address to the external world and connections from all devices on the home network appear to the external world as though they originate from a single IP address (that of the modem).

Task:

Some switches in your topology will be designated to be NAT switches. In each such switch, all but one of the ports will be connected directly to clients on the network. The last port will be dedicated to connect outside the network (uplink port). You are required to perform NAT operations when machines within the network communicate with machines outside the network.

For example, lets take the following list of clients on a network:

A • 172.168.12.15
B • 172.168.12.25
C • 172.168.12.32

Hosts A, B and C are all connected to switch S which in turn is connected to a router R via the interface which has a network address of 172.168.12.1 (called "private interface/network") and is the default gateway for A, B and C. R also has an uplink port (or multiple uplink ports) which is connected to other networks (called "public network" from here on) via an interface with an IP address 10.0.0.4.
When A makes a TCP request from port 10000 to an external address, say port 80 on 1.2.3.4:

- The packet is sent to R's interface with address 172.168.12.1.

- R determines that the packet is destined to the public network and the private IP address, port needs to be translated.
- R should create or reuse a translation entry in its Port Allocation Table (this is a 1 to 1 mapping between internal network IP, port number to public IP, port number). In this case, the translation entry can look like (172.168.12.15, 10000, 10.0.0.4, <external port>).
  External port number is randomly generated between number 1024 and 65535.
- So, a new packet is crafted with source address as the router's uplink IP address and source port as the translated port number.

When R receives a packet destined to its uplink IP address and port, p:

- It checks the Port Allocation Table to see if an translation entry exists for the port, p.
- If yes, update the destination IP address and port based on the matched translation entry and send it inside the network. ● If not, drop the packet.

Basic NAT functionality works only for flows that are originated from inside the NAT. Some protocols (such as FTP) subsequently attempt to establish flows from the remote server back to the initiating host. Unless the NAT has explicit support for these protocols, they will not work. You are welcome to extend your NAT to support FTP as well as any other protocols with similar behavior if you desire.

Further information regarding NAT can be found in the following links :

- http://en.wikipedia.org/wiki/Network_address_translation
- http://www.ipprimer.com/nat.cfm
- Network Address Translation on P&D p•335.